Final Report

Covering the Period from

1 May 1976 to 30 June 1982

014524-21-T

SEL 170

Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems

Principal Investigator

John F. Meyer

July 1982

(NASA-CR-170229) MODELS AND TECHNIQUES FOR EVALUATING THE EFFECTIVENESS OF AIRCRAFT COMPUTING SYSTEMS Final Report, 1 May 1976 - 30 Jun. 1982 (Michigan Univ.) 25 p HC A02/MF A01 CSCL 09B G3/60

N83-24184

Unclas 03435

Prepared for

National Aeronautics and Space Administration

Langley Research Center

Hampton, Virginia 23365

G. E. Migneault—NASA Technical Officer

NASA Grant NSG 1306

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SYSTEMS ENGINEERING LABORATORY

THE UNIVERSITY OF MICHIGAN, ANN ARBOR 48109

014524-21-T

SEL 170

MODELS AND TECHNIQUES FOR EVALUATING THE EFFECTIVENESS OF AIRCRAFT COMPUTING SYSTEMS

Final Report

Covering the Grant Period

1 May 1976 – 30 June 1982

Prepared for

National Aeronautics and Space Administration

Langley Research Center

Hampton, Virginia 23365

G. E. Migneault, NASA Technical Officer

NASA Grant NSG 1306

J. F. Meyer, Principal Investigator

Department of Electrical and Computer Engineering

Systems Engineering Laboratory

The University of Michigan, Ann Arbor 48109

July 1982

TABLE OF CONTENTS

1. INTRODUCTION

This report is the Final Report on the research project "Models and Techniques for Evaluating the Effectiveness of Aircraft Computing Systems" conducted for the NASA Langley Research Center under NASA Grant 1306. The subject grant was initiated 1 May 1976 for a one year period, extended 1 May 1977 for a second one year period, extended 1 June 1978 for a third one year period, extended 1 July 1979 for a fourth one year period, extended 1 July 1980 for a fifth one year period, and extended 1 July 1981 for a sixth one year period. This report summarizes work accomplished throughout the period of the grant, that is, the period from 1 May 1976 to 30 June 1982, hereafter referred to as the grant period.

The purpose of this research project was to develop models, measures, and techniques for evaluating the effectiveness of aircraft computing systems. By "effectiveness" in this context we mean the extent to which the user, i.e., a commercial air carrier, may expect to benefit from the computational tasks accomplished by a computing system in the environment of an advanced commercial aircraft. Thus, the concept of effectiveness involves aspects of system performance, reliability, and worth (value, benefit) which must be appropriately integrated in the process of evaluating system effectiveness. Specifically, the primary objectives of this project are:

I. The development of system models that can provide a basis for the formulation and evaluation of aircraft computer system effectiveness,
II. The formulation of quantitative measures of system effectiveness, and
III. The development of analytic and simulation techniques for evaluating the effectiveness of a proposed or existing aircraft computer.

During the first year of the project, a decision was made to decouple the performance and reliability aspects of effectiveness from the worth aspect, and to focus the effort on issues of performance and reliability. As argued when this research was originally proposed and as substantiated by research

accomplished to date, the issues of performance and reliability must be dealt with simultaneously in the process of evaluating the effectiveness of "degradable" computing systems. The term "performability" was introduced to refer to this unification of performance and reliability, and performability was identified with effectiveness in the statement of objectives I-III.

Research performed to date has made considerable progress toward the accomplishment of these objectives. During the first three years of the project [27], [31], [34]-[37] our effort was devoted primarily to the development of user-oriented methods wherein performance is represented by a discrete performance variable (DPV). In the fourth year [23], [24], work on refinements of the DPV methodology was accompanied by an initial investigation of design-oriented evaluation methods, where we seek closed-form solutions of continuous performance variables (CPV's) as well as DPV's.

In view of this progress and in keeping with future needs expressed by the NASA Langley Research Center, research proposed for the fifth year was more broadly conceived and had the following overall objective (which includes I-III above):

> The development of formal models and methods to aid the design and validation of fault-tolerant avionic systems.

During the fifth year [20], [21], investigation of design-oriented evaluation methods evolved into a major activity, balanced by a continued effort on refinements of the DPV methodology. The latter is aimed at taking the existing methodology to a point where it can be translated, with relative ease, into a programmed evaluation tool suitable for AIRLAB.

With the understanding that support, under the subject grant, would terminate after the sixth year (the current funding period), the research proposed for this year [38] was a continuation of the previous year's activity, in an effort

to reach a logical stage of completion for the overall project. Activity during the first half of this year was described in [16].

Section 2 of this report reviews the manpower effort proposed for the current year and lists the personnel involved in conducting the investigation, along with their levels of effort during the last six months of the grant. Section 3 summarizes the research performed during the grant period.

2. PERSONNEL

In the proposal for the current year [38], it was estimated that the following effort would be required.

Principal Investigator

50%, July 1981
100%, August 1981
20%, September - December 1981
10%, January - May 1982

Two Graduate Student Research Assistants

50%, July - August 1981
25%, September - December 1981

Secretary

25%, twelve months, calendar year.

During the six month period from 1 January 1982 to 30 June 1982, personnel and their levels of effort have been as follows.

Principal Investigator

John F. Meyer:	20%, January-May 1982

Secretary

Virginia Folsom:	25%, January - June 1982

3. TECHNICAL STATUS

The following sections briefly describe the technical status of the research conducted during the grant period. The descriptions presented here are described in more detail within the appropriate Semi-Annual Status Reports or Technical Reports.

3.1. Review and Assessment of Related Work

During the first three months of the project, we reviewed and assessed related work bearing on the objectives of the project. This work is described in [37] and included an assessment of traditional structure-based reliability models in an effort to indicate how such models might be generalized to provide a basis for the formulation and evaluation of system effectiveness.

3.2. System Requirements, Missions, and Tasks

In order to practically evaluate system effectiveness, it is essential to have an understanding of the user's desired object system goals, in consonance with the requirements, constraints, and interface characteristics of the "world" in which it will ultimately operate. Early in the grant, we devoted some effort towards delineating system requirements, constraints, etc., associated with the use of aircraft computers by commercial air carriers. Efforts in this direction have been initiated by others (see [39], [40], for example) and we attempted to build on these existing views as much as possible. The purpose of this activity was not to obtain a set of system specifications, per se, but, instead, to obtain appropriate informal descriptions of system behavior at various levels of abstraction. The results of this activity are described in [37].

3.3. Development of System Models

In parallel with our efforts to develop informal descriptions of system behavior (see section 3.2), we initiated during the first year the development

of a formal model hierarchy whose levels of abstraction correspond to informal descriptions at the mission level, functional task level, and computational task level. The bottom level of the hierarchy corresponds to low level descriptions of the computer's hardware and software.

We seek a model of the total system with a behavior relating directly to the user's requirements and a structure accurately describing the probabilistic nature of the system's components. This view requires a high, user-oriented level with scope comprising the total system (ie., the air carrier) as well as a low, structure-oriented bottom level comprising the object system (i.e., the computing system and closely related peripheral equipment).

In order to relate the performance of the computer hardware (bottom level) to the accomplishment of user-oriented missions (top level), intermediate levels may also be necessary. Because the bottom level concerns the object system, we have found that information from non-object systems (e.g, environment, supporting, and related systems) may be more easily introduced at these intermediate levels. Using what we call "basic variables," we can incorporate each non-object system into the hierarchy based on the level at which that information is used. For example, "weather" does not depend on any aircraft function and yet it can affect the mission outcome; thus, weather may be introduced at the aircraft functional level.

The bottom model, along with the higher level basic variables, are referred to collectively as the "base model" of the total system. Formally, the connection between the behavior of the base model and that of the top (mission) level expressed by a "capability function" γ. In general, the interaction between various levels of a model hierarchy can be viewed either as part of the hierarchy, per se, or as something with is is determined later, in the process of using the model to analyze some aspect of system behavior, e.g., its perfor-

mability. Either view is legitimate, but the latter appears to be more convenient for the purpose of classifying and discussing these interactions. In [36] we introduced these concepts and, for the case of a discrete set of accomplishment levels, developed some simple descriptions of higher level models, along with some stochastic models that can serve as bottom level models in the hierarchy. In [35] we developed a probability-theoretic basis for the modeling framework discussed in [36], [37]. This formal representation permits us to rigorously state various intuitive concepts and assumptions associated with models of the total system. It also provides us with a more precise foundation for the investigation of model simplification techniques such as time "phasing" and state "lumping." Early work on this problem was presented at [11]; the formal basis for the modeling framework was published in [9], presented at [14]-[16] and further refined in [28] and [2].

3.4. Phased Models

One approach to dealing with a time-varying environment is to decompose the system's utilization period into consecutive time periods (usually referred to as a decomposition of the system's "mission" into phases; see [41]-[43], for example). Demands on the system are then allowed to vary from phase to phase; within a given phase, however, they are assumed to be time-invariant. This permits intraphase behaviors to be evaluated in terms of conventional time-homogeneous models, but raises the interesting question of how the intraphase results are combined. This is the essential question addressed in investigations of "phased mission" reliability evaluation methods (e.g., [41]-[43]) where the problem has been constrained as follows. It is assumed, first, that a "success criterion" (formulated, say, by a "structure function" see [43] for instance) can be established for each phase, where the criterion is independent of what occurs during other phases. It is required further that

successful performance of the system be identified with success during all phases, that is, the system performs successfully if and only if, for each phase, the corresponding success criterion is satisfied throughout that phase.

Although the above constraints are reasonable for certain types of systems, they exclude systems where successful performance involves nontrivial interaction among the phases of the mission. In more exact terms, it has been shown (see [10], Theorem 6) that such "structure-based" formulations of success are possible if and only if the phases are "functionally independent" in a precisely defined manner. What we have done, therefore, is examined the utility of "phased models" in a less restricted context.

In addition to removing the above constraints, we have extended the domain of application to include evaluations of computing system performability. Finally, unlike the models used in phased mission reliability evaluatian, we permit the state sets of the intraphase models to differ from phase to phase. Thus, the modeling of a particular phase can be tailored not only to the computational demands of that phase but also to the relevant properties of the total system which influence performance during that phase. We investigated phased base models in [7], [30].

3.5. Generalized Phased Models

In the context of our current discrete performance variable (DPV) methodology, phased models (see Section 3.4) play a central role in that they permit the capability function to be formulated in terms of a discrete-time stochastic process $\bar{X}$ derived from a continuous-time base model X. As defined and investigated in [7], [30], a phased base model $\bar{X}$ is obtained form X by essentially sampling the intraphase processes at the ends of their respective phases. Such models suffice when there are no cycles in the state-transition-rate diagrams of the intraphase processes (see the evaluation of SIFT [3], [6], [33],

for example). On the other hand, if there is a non-zero probability of entering a previously visited state (e.g., when recovering from a transient fault or a software error), then an end-of-phase sample may no longer reflect the intraphase behavior.

To rectify this deficiency, we have investigated a more general notion of phasing wherein the random variable associated with phase k is a "summary" of the system's behavior during phase k. More precisely, the k^{th} intraphase model is regarded as a performability model in its own right, where the performance variable (denoted Y_k) is the variable that summarizes the intraphase behavior during phase k. Assuming m phases, the set of variables $\bar{X} = \{Y_1, Y_2, \ldots, Y_m\}$ then constitutes a discrete-time model on which the formulation of the capability function is based.

Study of these generalized phased models involved two principal activities. The first concerns formulating capability functions via special types of "organizing functions" so as to facilitate the solution of trajectory sets (of the process $\bar{X}$); see [17]. The second area concerns how the probabilistic nature of $\bar{X}$ (i.e., the probability distributions of the variables Y_k) is determined. Here we invoke the concept of a functional of a stochastic process (see [44], for instance). To determine the performability of each intraphase performability model, we have developed solution techniques involving Markov renewal theory and Laplace transform methods. The approach permits us to express the solutions in terms of matrix representations which can then be applied to the iterative formulas developed to evaluate the performability of the phased model.

This work is documented in [5], [17], [25].

3.6. Functional Dependence

In system analysis, the concept of dependence among subsystems is often based on their physical interconnections. However, subsystems may also depend on one another as they cooperate in the realization of some specified level of system performance. Such dependence is referred to as "functional," where dependent objects may be distinguished in time as well as space. e.g., a subsystem observed at one time may functionally depend on itself (or on some other subsystem) observed at another time. The need for a general concept of functional dependence arises in the context of performability evaluation. Questions about the nature, properties, and use of functional dependence were studied and reported upon in [10], [35]-[37] and extended in [34].

Classically, when one looks for the dependencies between subsystems, it is in the hope that the subsystems under consideration will turn out to be independent. In this case, each subsystem can then be studied separately. However, not all forms of dependency necessarily complicate the analysis. For instance, if one knows that subsystem S_2 "totally" depends on subsystem S_1, that is if knowing the state of S_1 yields all the relevant information about S_2, then one may essentially disregard S_2 when analyzing the total system. In particular, such simplifications are often made in evaluations of system reliability.

In [10], we considered functional dependence between system coordinates where, generally, a given coordinate represents some specified part (subsystem) of the system observed at some specified point in time. This set (D) of system coordinates was assumed to be finite. Dependence was defined relative to a "structure set" R where R is a subset of the Cartesian product set determined by the system coordinates. (Because of the central role of the set R, we sometimes refer to functional dependence as "R-dependence.") In [34], we

investigated functional dependence when the index set D is countably infinite.

Using the basic functional dependence theorems, we established the fundamental limitations of reliability modeling that is based on "structure functions" or, equivalently, their representation by "fault trees." In particular, we showed that any phased system model, wherein the capability function can be described by a sequence of structure functions (fault-trees), is characterized by a total absence of functional dependence among the phases (where the dependence is relative to the set of all state trajectories corresponding to system "success"). One of the features of performability modeling, on the other hand, is its ability to accommodate interphase dependencies.

3.7. Hierarchical Modeling of Air Transport Missions

Several prototype air transport models were examined in the course of the grant period. These models are described in detail in [3], [6], [8], [34]-[37]. Many of these models are comprehensive examples and illustrate some of the concepts discussed in the previous sections. We also initiated an ambitious modeling project of the FTMP computer [45], [46]; see [20], [21], [23], [24].

3.8. Evaluation Algorithms and Programs

Concurrent with the development of performability models, concepts, measures, and measure formulations, we also initiated the development of evaluation algorithms [20], [21], [23], [24], [34], [35]. As an implementation of these algorithms, we also began development of prototype tools for the purpose of investigating design issues. These tools were incorporated into the software package called METAPHOR (Michigan Evaluation Aid for PerpHORmability, [29], [32]. Since its inception, METAPHOR has progressed through several implementations. The earliest version took as input the base model

trajectory sets for each accomplishment level and information about the probabilistic nature of the base model; from this data, METAPHOR calculated the system's performability. However, obtaining the base model trajectories is generally difficult and so the later versions automated to a great extent this portion of the modeling.

The algorithms in which we were particularly interested are those for calculating the base model trajectory set U_a associated with an accomplishment level a. The goal here was to automate those tasks which are mechanical, laborious, and error-prone. These tasks include:

(1) Calculating the inverse image $\gamma_{i+1}^{-1}(a)$ i.e., the set of all base model state trajectories at level i+1 that correspond to an accomplishment level a, given the inverse image $\gamma_i^{-1}(a)$.
(2) Finding a minimal representation (in terms of the number of array products; see [35], p. 96) of trajectory sets,
(3) Checking that all trajectories have been included for each "coordinate inverse" of the interlevel translation κ_{i+1}, and if some of those trajectories have not been so included, determining which have been excluded, and
(4) Allowing input of non-mutually exclusive trajectory sets.

During the reporting period, work was completed on the implementation of an algorithm for items (3) and (4). An algorithm for item (1) was designed and partially implemented: METAPHOR can now evaluate our earlier examples (e.g., the somewhat complex example of [35]) with no difficulty, and can proceed to a significant depth (to the last level) with the evaluation of the SIFT example [34]. Regarding item (2), criteria of representation efficiency, other than the number of array products, were investigated. Because we are dealing with computational algorithms, "efficiency" relates to both

1) the amount of space required to represent the functions, and

2) the amount of time required to determine the representations of those functions.

We investigated some of the space and time tradeoffs for computing γ^{-1} [20].

3.9. Closed-form Models and Solutions

Our work on the derivation of closed-form performability solutions was motivated by design considerations and, specifically, by the need to support design-oriented validation. Efforts dealing with each of these needs have been pursued during the reporting period, with the emphasis placed on design-oriented validations.

If a performability evaluation indicates that a system design is valid, i.e., the system satisfies its performability specification, then the evaluation has served its purpose (This is not to say that this phase of the validation process is complete; other validation methods, both formal and informal, must be invoked so as to establish greater confidence in the design's validity.) If, on the other hand, the results of a performability evaluation disclose that a design is deficient, the performability data need not be indicative of just how the design should be modified. This is due to the fact that lower level, design-oriented details are often suppressed by a user-oriented performance variable. Hence early validation (during the design process) at lower system and subsystem levels is required if negative results are to indicate how the design should be modified. In the latter validation context, and more generally, in the context of "design aids," performability models and solutions can likewise play an important role. To support the investigation of various design tradeoffs, we investigated various methods which yield parametric performability solutions, expressed in terms of various system and environmental parameters.

Generally, the difficulties encountered in parametric evaluation are due to the fact that performability must be formulated directly in terms of performance levels, thereby restricting the mathematical nature of the capability

function. To compensate for these restrictions, one seeks methods for representing underlying variations (at the base model level) in a form that matches constraints imposed by the capability function. Another strategy, which can be applied simultaneously, is to relax these restrictions via innovative decompositions of the capability function and the solution procedure.

We began our investigations by studying a degradable dual-processor with an input buffer (queue) for the temporary storage of computational tasks that arrive randomly at the input. To solve this system, we extended the kind of Markovian queueing models that are currently employed to evaluate the performance of a (fault-free) computer (see [47], [48], for example). When so extended, these models are able to represent variations in structure, due to faults, as well as variations in internal state and environment. In solving the performability, our strategy is to lump states of the base model so that, within a lump, the model exhibits a steady-state behavior (to a close approximation). This permits decomposition of the solution into an equilibrium (steady-state) part and a transient part. The equilibrium part employs techniques that typically are used in solving queueing models; the transient part is more difficult and calls for innovative extensions of known techniques. Here, through a hierarchical decomposition of the capability function and an appropriate partitioning of the accomplishment set, we are able to obtain the desired solution. Our initial work is described in [26].

We further extended these results to the modeling of a degradable buffer/multi-processor system with N processors. In the context of this generalized example, we were forced to develop a more systematic solution procedure (for the transient part of the solution) that could be feasibly applied to a system with more than two processors. Various solution approaches were considered, including a recursive formulation patterned after a formulation proposed by Howard (see [49], p.861) for determining the "expected value" of

a similar type of performance variable. What we seek, however, is the complete probability distribution function of Y (not just its expected value E[Y]) and, when so formulated, we were unable to find a feasible means of solving the equations. (Even in the case of expected values, the Howard formulation does not appear to yield a practical means of solution.) The solution procedure we finally adopted was a natural extension of that used in the two processor case discussed in [26]. The results of this effort were presented at [13] and were documented in [1], [4], [22].

The algorithm that we developed delineates in broad terms the basic method for arriving at solutions. We have further investigated suggest specific techniques for actually carrying out the prescribed steps. In particular, the regions of integration $C_y = \gamma_1^{-1}(B)$ (see [22], p. 22) must be characterized. Thus, the computational example presented in [4] was derived in a relatively *ad hoc* manner; effectively, the solution was based on a graphical argument. Such an approach becomes more difficult when the number of servers is three and becomes intractable when the number of servers grows to four or more. In [20], we presented an integral solution for the class of systems having the single state trajectory $(m, m-1, \ldots, 0)$. The crux of the solution is the characterization of the regions C_y. We have also solved examples where the underlying operational model is *not* Markov. For instance, we have examined systems where failure rates are dependent on the history of the system; see [18].

3.10. Stochastic Modeling of Parallel Systems

This work was motivated by our concern with modeling complex integrated systems such as avionic systems where, due to additional complexity (as compared, say, with an aircraft computer), representation must take place at higher (less detailed) levels of abstraction. When represented at such levels, a system will typically exhibit a greater amount of parallelism and/or

nondeterminacy. Parallelism and nondeterminacy are important properties of complex systems which have been studied in a variety of contexts. There is a lack, however, of universal definitions which clearly distinguish these notions; in most cases, they are either intermixed or viewed as the same. One reason is that most existing models of parallel systems (e.g. Petri nets [50]), fail to distinguish nondeterminism due to parallelism from nondeterminism due to uncertainty in the consequences of an action (we refer to the latter as "nondeterminacy"). In Keller's concept of a "named transition system" [51], nondeterminacy can be distinguished in certain cases (i.e., when two or more transitions from the same state have the same name), but not in all cases.

To remedy this deficiency, our work has included formulation of a class of general models, called *dynamic transition systems* (DTS's), wherein parallelism and nondeterminacy can be clearly distinguished; see [18]. DTS's represent system state-behavior at the same level of abstraction as Keller's named transition systems, but are more general in that the "enabling" of transitions is no longer tied to the transition relation. (In a named transition system, a transition t is "enabled" in state q if and only if there is a state transition from q named t.) Instead, we allow the set of enabled transitions to be one of a specified set of alternatives, thereby introducing a source of nondeterminacy that has useful interpretations and is easily distinguished from parallelism. Moreover, this same distinction can be captured in lower level (more detailed) network models, e.g., a class of models called *dynamic P-nets* (DPN's) which constitute an analogous generalization of (ordinary) Petri nets; see [18].

Our principal objective in defining DTS's and DPN's was to provide a more suitable point of departure for formulating stochastic versions of these models. Accordingly, our concept of a *stochastic transition system* (STS) is defined as the stochastic extension of a DTS; likewise, a *stochastic P-net* (SPN)

is the stochastic extension of a DPN.

The modeling power of STS's and SPN's is quite extensive and includes, for example, all systems that can be modeled by Markovian queueing models. The latter, however, are restricted in their ability to represent various forms of parallelism and nondeterminacy. Different models have been proposed to overcome these deficiencies [52]-[59] but, with the exception of [59] these models appear to have limited applicability. On the other hand, the type of stochastic Petri nets proposed by Natkin [59] are better suited to our needs and provided a stimulus for our current research. There remained, however, the problem of distinguishing parallelism from nondeterminacy, since the models of [59] are stochastic extensions of (ordinary) Petri nets. This precipitated the development described above and, specifically, led to formulation of stochastic P-nets (SPN's). By their construction, SPN's are more general than Natkin's stochastic Petri nets (hence our use of the name P-net). Moreover, this added generality is indeed very useful in the context of performability evaluation.

Concerning the modeling power of STS's, we have obtained some interesting results which include the following. For the case when the processing periods of the processes (transitions) are exponentially distributed, and there are certain independence properties in the processing periods of different processes (transitions) and in the behavior of nondeterminacy in the system, it turns out that the state behavior of the system can be modeled as a time-homogeneous semi-Markov process. We have also obtained a closed-form solution for the corresponding semi-Markov kernel. This result is especially important because this stochastic process can be used directly as a base model for performability evaluation. Another result is related to the priority and interactions among the processes (transitions) of the system. We have found that the Markovian property of the state behavior is independent of a

rich class of the priority types and interactions among the processes (transitions) of the system. This constitutes a generalization of related results in queueing theory. Results from this research were presented at [12].

3.11. Bibliography on Formal Methods

During the reporting period, we conducted a search of recent literature concerning formal methods for system specification, design and validation. The aim of this search was to classify current literature on formal methods that might be meaningfully exploited in the specification, design, and validation of avionic systems (where validation includes verification, testing and evaluation). The specific literature searched includes journal papers, conference papers, and technical reports published during the five years from 1977 to 1981. The articles are classified according to five topic areas: specification, design, verification, testing and evaluation. Because the survey was completed in September 1981, no citations appearing after that date are included in the resulting bibliography [19].

4. PUBLICATIONS

4.1. Journal Articles

[1] J. F. Meyer, "Closed-form solutions of performability," *IEEE Transactions on Computers*, July 1982, pp. 648-657.

[2] J. F. Meyer, "On evaluating the performability of degradable computing systems", *IEEE Transactions on Computers*, August 1980, pp. 720-731.

[3] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Performability evaluation of the SIFT computer", *IEEE Transactions on Computers*, June 1980, pp. 501-509.

4.2. Conference Papers

[4] J. F. Meyer, "Closed-form solutions of performability," in *Proc. 1981 Int'l Symposium on Fault-Tolerant Computing*, Portland, ME, June 1981, pp. 66-71.

[5] J. F. Meyer and L. T. Wu, "Evaluation of computing systems using functionals of a Markov process", *Proc. 14th Hawaii Int'l Conf. on System Sciences*, Honolulu, HI, Jan. 1981, pp. 74-83.

[6] J. F. Meyer, D. G. Furchtgott, L. T. Wu, in "Performability evaluation of the SIFT computer", in *Proc. 1979 Int'l Symp. on Fault-Tolerant Computing*, Madison, WI, pp. 43-50, June 1979.

[7] J. F. Meyer and L. T. Wu, "Phased models for evaluating the performability of computing systems," in *Proc. 1979, Conference on Information Sciences and Systems*, The John Hopkins Univ., Baltimore, Maryland, March, 1979.

[8] J. F. Meyer and D. G. Furchtgott, "Performability evaluation of fault-tolerant multiprocessors," in *Digest of 1978 Government Microcircuit Applications Conference*, Monterey, California, November 1978, pp. 362-365.

[9] J. F. Meyer, "On evaluating the performability of degradable computing systems," in *Proc. 8th International Symposium on Fault-Tolerant Computing*, Toulouse, France, June 1978.

[10] J. F. Meyer and R. A. Ballance, "Functional dependence and its application to system evaluation," in *Proc. 1978 Conference on Information Sciences and Systems*, The Johns Hopkins University, Baltimore, MD, March 1978.

[11] J. F. Meyer, "A model hierarchy for evaluating the effectiveness of computing systems," in *Proceedings 3rd National Reliability Symposium*, Perros-Guirec, France, September 1976, pp. 539-555.

4.3. Papers Presented (Not Published)

[12] J. F. Meyer, "Performance-reliability evaluation of parallel systems", presented at the IEEE Computer Society Workshop on the Reliability of Local Area Networks, South Padre Island, TX, Feb. 1982.

[13] J. F. Meyer, "Closed form solutions of performability," presented at the Workshop on the Validation of Fault-Tolerant Computers and Systems (IEEE), Luray, VA, Sept. 1980.

[14] J. F. Meyer, "Unified performance-reliability evaluation of degradable computing systems," presented at the IFIP Working Conference on Reliable Computing and Fault-Tolerance, London, England, Sept. 1979.

[15] J. F. Meyer, "Evaluating the unexpected," presented at the Workshop on Designing for the Unexpected (IEEE), St. Thomas, Virgin Islands, December, 1978.

[16] J. F. Meyer, "Modeling concepts for unifying performance and reliability evaluation," presented at the Symposium on Modelling and Simulation Methodology, Rehovot, Israel, August 1978.

4.4. Technical Reports

[17] L. T. Wu, "Models for evaluating the performability of degradable computing systems," Systems Engineering Laboratory Technical Report No. 169 The University of Michigan, Ann Arbor, June 1982.

[18] J. F. Meyer, D. G. Furchtgott, and A. Movaghar, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 164, The University of Michigan, Ann Arbor, January 1982.

[19] J. F. Meyer, D. G. Furchtgott, and A. Movaghar "A bibliography on formal methods for system specification, design, and validation," Systems Engineering Laboratory Technical Report No. 163, The University of Michigan, Ann Arbor, January 1982.

[20] J. F. Meyer, D. G. Furchtgott, and A. Movaghar, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 155, The University of Michigan, Ann Arbor, July 1981.

[21] J. F. Meyer, D. G. Furchtgott, A. Movaghar, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 148, The University of Michigan, Ann Arbor, January 1981.

[22] J. F. Meyer, "Closed-form solutions of performability," Systems Engineering Laboratory Technical Report No. 147, The University of Michigan, Ann Arbor, January 1981.

[23] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 145, The University of Michigan, Ann Arbor, July 1980

[24] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 141, The University of Michigan, Ann Arbor. January 1980.

[25] J. F. Meyer and L. T. Wu, "Evaluation of computing systems using functionals of a stochastic process," Systems Engineering Laboratory Technical Report No. 140. The University of Michigan, Ann Arbor, July 1980.

[26] J. F. Meyer, "Performability models and solutions for continuous performance variables," Systems Engineering Laboratory Technical Report No. 139, The University of Michigan, Ann Arbor, July 1980.

[27] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 138, The University of Michigan, Ann Arbor, July 1979.

[28] J. F. Meyer, "Performability modeling with continuous accomplishment sets," Systems Engineering Laboratory Technical Report No. 137, The University of Michigan, Ann Arbor, July 1979.

[29] D. G. Furchtgott, "METAPHOR (Version 1) User's Guide," SEL Report No. 136, Systems Engineering Lab, The University of Michigan, Ann Arbor, MI, July 1979.

[30] L. T. Wu and J. F. Meyer, "Phased models for evaluating the performability of computing systems," Systems Engineering Laboratory Technical Report No. 135, The University of Michigan, Ann Arbor, July 1979.

[31] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 129, The University of Michigan, Ann Arbor, January 1979.

[32] D. G. Furchtgott, "METAPHOR (Version 1) Programmer's Guide," SEL Report No. 128, Systems Engineering Lab, The University of Michigan, Ann Arbor, MI, Jan. 1979.

[33] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Performability evaluation of the SIFT computer," SEL Report No. 127, Systems Engineering Lab, The University of Michigan, Ann Arbor, MI, Jan. 1979.

[34] J. F. Meyer R. A. Ballance, D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 121, The University of Michigan, Ann Arbor, July 1978.

[35] J. F. Meyer R. A. Ballance, D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 116, The University of Michigan, Ann Arbor, January 1978.

[36] J. F. Meyer R. A. Ballance, D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems Engineering Laboratory Technical Report No. 111, The University of Michigan, Ann Arbor, July 1977.

[37] J. F. Meyer D. G. Furchtgott, and L. T. Wu, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Systems

Engineering Laboratory Technical Report No. 106-1, The University of Michigan, Ann Arbor, MI, Nov. 1976.

5. REFERENCES

[38] J. F. Meyer, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Proposal for extension of NASA Grant NSG 1306, submitted to NASA Langley Research Center, May 1981.

[39] R. S. Ratner, E. B. Shapiro, H. M. Zeidler, S. E. Wahlstrom, C. B. Clark, and J. Goldberg, "Design of a fault-tolerant airborne digital computer," NASA Contract NAS1-10920, vol. II, Final Report, Stanford Research Institute, Oct. 1973.

[40] B. E. Bjurman, G. M. Jenkins, C. J. Masreliez, K. L. McClellan, and J. E. Templeman, "Airborne advanced reconfigurable computer (ARCS)," NASA Contract NAS1-13654, Boeing Commercial Airplane Company, Seattle, WA, Aug. 1976.

[41] H. S. Winokur, Jr. and L. J. Goldstein, "Analysis of mission-oriented systems," *IEEE Trans. Reliability*, vol. R-18, no. 4, Nov. 1969.

[42] J. L. Bricker, "A unified method for analyzing mission reliability for fault tolerant computer systems," *IEEE Trans. Reliability*, vol. R-22, no. 2, June 1973.

[43] J. D. Esary and H. Ziehms, "Reliability of phased missions," in *Reliablity and Fault Tree Analysis*. Philadelphia, PA: SIAM, pp. 213-236, 1975.

[44] K. L. Chung, *Markov Chains with Stationary Transition Probabilities*. Berlin, Germany: Springer-Verlag, 1960.

[45] T. B. Smith, et al., "A fault tolerant multiprocessor architecture for aircraft," vols I-III, Technical Report, NASA Contract NAS1-13782, The Charles Stark Draper Laboratory, Inc, Cambridge, MA, July 1976, April 1977, and Nov. 1978.

[46] A. L. Hopkins, Jr., T. B. Smith, III, and J. H. Lala, "FTMP--A highly reliable fault-tolerant multiprocessor for aircraft," *Proc. of the IEEE*, vol. 66, no. 10, pp. 1221-1239, Oct. 1978.

[47] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology*. Reading, MA: Addison-Wesely, 1978.

[48] A. O. Allen, *Probability, Statistics, and Queueing Theory--With Applications*. New York, NY: Academic Press, 1978.

[49] R. A. Howard, *Dynamic Probabilisitc Systems, Vol. II: Semi-Markov and Decision Processes*. New York, NY: Wiley, 1971

[50] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[51] R. M. Keller, "Formal verification of parallel programs," *CACM*, vol. 19, pp. 371-384, July 1976.

[52] Y. W. Han, "Performance evaluation of a digital system using a petri net-like approach," in *Proc. National Electronics Conf.*, vol. 32, pp. 166-178, Oct. 1978.

[53] J. L. Baer and J. Jensen, "Simulation of large parallel systems: Modeling of tasks," in *Measuring, Modeling, and Evaluating Computer Systems*, H. Bellner and E. Gelenbe, Ed. Amsterdam, Netherlands: North-Holland, 1977.

[54] G. J. Nutt, "The formulation and application of evaluation nets," Thesis, Comp. Sci. Group, Univ. of Washington, Seattle, WA, July, 1972.

[55] J. D. Noe, "Nets in modeling and simulation," in *Net Theory and Application*. Berlin, Germany: Springer-Verlag, Springer Lecture Notes in Comp. Sci. No. 84, 1980.

[56] S. Shapiro, "A stochastic Petri net with applications to modelling occupancy times for concurrent task systems," *Networks*, vol. 9, pp. 375-379, Winter 1979.

[57] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets," Ph.D. thesis, Project MAC, MAC-TR-120, Cambridge, MA, Feb. 1974.

[58] J. Sifakis, "Performance evaluation of systems using nets," in *Net Theory and Application*. Berlin, Germany: Springer-Verlag, Springer Lecture Notes in Comp. Sci. No. 84, 1980

[59] S. Natkin, "An evaluation CAD tool based on stochastic Petri nets," 2nd Advanced Course, Computing Systems Reliability, SURF, Toulouse, France, Sept. 1979..